INFORMATION AND CONTROL 61, 1–14 (1984)

Dynamic, First-Fit Packings in Two or More Dimensions

E. G. COFFMAN, JR., AND E. N. GILBERT

Bell Laboratories, Murray Hill, New Jersey 07974

A rectangular storage area or *bin*, of width w and height h, stores nonoverlapping square objects, of sizes up to $k \times k$, that arrive and depart in an unpredictable sequence. Squares packed at any given time never exceed mw in total area. How large must h be to ensure that there is room for each square when it arrives? This problem generalizes a 1-dimensional packing problem, considered by Robson and others as a model of storage allocation in a computer. All packing algorithms considered here pack the new arrival on the lowest possible level. For all algorithms and all sequences of arrivals and departures, the required bin height h is shown to have an upper bound of the form $O(m \log k)$. Also, heights greater than a lower bound, also of form $O(m \log k)$, are actually needed for certain worst-case sequences. These bounds contain multiplicative constants that differ by a factor slightly less than 9. Numerical results show that the factor can be reduced to about 1.7. Similar results hold for packings of cubes, of maximum side k, into a rectangular parallelepiped in space of dimension $d \geqslant 3$.

1. INTRODUCTION

We consider packing problems in two or more dimensions under the assumptions: (i) requests to pack or store objects in a given space (or bin) arrive at unpredictable times, (ii) packed objects depart after unpredictable time intervals, and (iii) objects once packed cannot be moved prior to their departure. Under these assumptions the bin can become wastefully fragmented. That is, although there is sufficient total available space for a new arrival, there may be no one "hole" large enough to contain the object.

One-dimensional fragmentation problems arise naturally in the study of dynamic storage allocation in computers [1, 3]. There the bin is a linear memory and the objects are records or files stored in sequential locations. Similar fragmentation problems in two and three dimensions can arise in warehousing applications, where interest centers on estimates of the bin size needed and on efficient packing algorithms.

The 1-dimensional problem of packing intervals on a line, as originally posed by McIlroy [4], is known as the *bay restaurant problem.* Maxima were set on both the allowed length of each packed interval and on the allowed total length of all intervals packed at the same time. Robson [5, 6, 7] and others (see [1] for a full bibliography) then obtained bounds on the required

0019-9958/84 $3.00

bin length. In [2] the bin was replaced by several identical units. Section 2 generalizes the bin to a rectangle and the intervals to squares. The packing algorithms are all "first-fit" algorithms (higher-dimensional problems differ from 1-dimensional ones in allowing many first-fit algorithms instead of just one). Results for $d \geqslant 3$ dimensions closely parallel the 2-dimensional case. Section 3 gives lower bounds on the worst-case performance of first-fit rules. A corresponding upper bound is proved in Section 4. Subsequent sections tighten these bounds.

2. Packing Algorithms

Our problem in two dimensions concerns the packing of squares into a rectangle (or bin) of width w in the plane. An $i \times i$ packed square will be called an *i-square*. Both i and w will be integers. The rectangle will be imagined cut into unit square cells; for any integers x and y with $1 \leqslant x \leqslant w$ and $1 \leqslant y \leqslant \infty$, the unit square *located at* (x, y) is that unit square whose upper right corner has coordinates (x, y). The packed i-squares are always aligned to occupy i^2 whole cells. The *level* of a cell located at (x, y) is defined to be y. An i-square is said to be *at* level y if y is the maximum of the levels of the cells it occupies.

The packing problem is dynamic; i.e., each square arrives at a particular time and remains in the packing until a specified departure time. L will denote a list of square sizes, arrival times, and departure times. For a given L, a packing algorithm must decide where to pack each square so that different packed squares have no cell in common. Once a square is packed it is never moved to a new location to make room for new squares. We restrict ourselves to *first-fit* algorithms, i.e., algorithms that pack each square at the lowest level available at the square's time of arrival.

A newly arrived square may fit on the lowest level in several positions. Hence there are many first-fit algorithms. The *left-justifying algorithm* places the new square as far to the left as possible on the lowest level. For a given algorithm A, $A(L)$ will denote the maximum level ever reached by any square packed from the list L. Left-justifying achieves reasonably dense packings but is not a best algorithm for all lists L. For example, with $w = 5$, let a 1-square, 2-square, and 3-square arrive in that order. The left-justifying algorithm has $A(L) = 5$. Other first-fit algorithms, that leave a gap of 2 units between the 1-square and 2-square, have $A(L) = 4$.

The main problem will be to bound $A(L)$ for all first-fit algorithms and lists L that contain squares no bigger than $k \times k$. The maximum total area of squares from L that are ever packed simultaneously, to be denoted mw, is an important parameter of L. Although $A(L)$ is approximately m when squares are always packed "perfectly," i.e., with no unoccupied cells at low levels, the following examples show that $A(L)$ can be much larger than m.

3. Examples

Explicit examples will provide lower bounds on the levels $A(L)$ attained by lists L with squares that are k by k or smaller. All examples have the same general form. The squares have only special sizes, say s_0-squares, s_1-squares,..., s_K-squares with $1 \leqslant s_0 < s_1 < \cdots < s_K \leqslant k$. For each a, with $0 \leqslant a \leqslant K$, the s_a-squares arrive together as an event A_a. These events occur in the order $A_0, A_1, \ldots, A_K$. Between events A_a and A_{a+1} some of the packed s_i-squares $(0 \leqslant i \leqslant a)$ depart, an event called B_a. L should prescribe many departures at B_a, but without creating a hole large enough for an A_{a+1}-square. There can then be many arrivals at A_{a+1} and all must be packed at higher levels.

An *s-row* is a set of all s-squares that have the same y coordinate. A *complete* s-row is an s-row that leaves no gap of size s or more (into which another s-square could be packed). The number of s-squares in a complete row depends on the algorithm A. There can be at most $[w/s]$ squares, a number achieved by left-justifying. For any A, a complete row must contain at least $(w - s + 1)/(2s - 1)$ s-squares. The lists in this section will always prescribe a number of arrivals at A_a that fills an integer number r_a of complete s_a-rows with no s_a-squares left over.

Theorem 1. *Let A be the left-justifying algorithm. Let k be a power of 2. Let w be divisible by k and m by $4k$. Then a list L exists with*

$$A(L) = m(1 + 0.75 \log_2 k).$$

Proof. Let $k = 2^K$ and construct a list, as described above, using squares of sizes $s_a = 2^a$ for $0 \leqslant a \leqslant K$. The proof, which extends a one-dimensional result of Robson [7], will only be sketched. Figure 1, with $k = 4$, $m = w = 16$, and $A(L) = 40$, illustrates the idea. At A_0, mw 1-squares arrive, filling the $r_0 = m$ complete rows with $y = 1,\ldots, m$. At B_0, $\frac{3}{4}$ of these squares depart, leaving only squares at cells with x and y both even. Now a total area $3mw/4$ is available for 2-squares that arrive at A_1. They cannot fit on rows $1,\ldots, m$ but completely fill a new band of rows $m + 1 \leqslant y \leqslant m + 3m/4$. At B_{a-1}, all previously packed s_i-squares $(i = 0, 1,\ldots, a - 1)$ depart except those at squares with both x and y divisible by $s_a = 2^a$. One can verify by induction that these departures again have total area $3mw/4$. Then 2^a-squares of A_a completely fill a new band of $r_a s_a = 3m/4$ rows. After A_K the k-squares reach rows as high as $m + (3m/4)K$, proving the theorem. ∎

Because w was assumed divisible by k, and hence by s_a, the s_a-rows always covered all w columns neatly with no gaps at the end. Likewise, assuming $4k$ divides m made the area $3mw/4$ neatly divisible into complete 2_a-rows. The next result removes these assumptions.

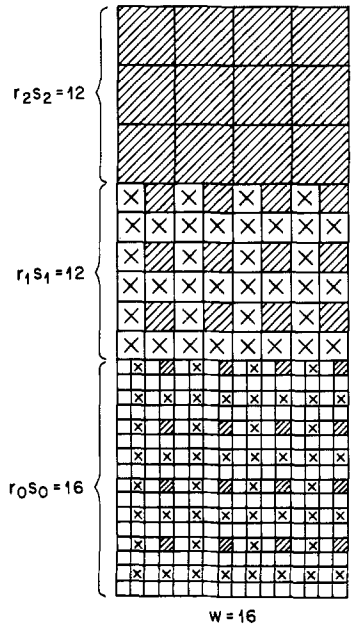

FIG. 1. Packing with $w = m = 16$ after the arrival A_2 of 4-squares. Shaded cells are occupied. Squares marked with an X departed at B_1. Unmarked squares departed at B_0.

COROLLARY. *Let K denote $[\log_2 k]$ and suppose $2^K \leqslant w$. Lists L using squares no larger than k by k can achieve levels*

$$A(L) \geqslant (4 + 3K)\, 2^K \left[\frac{2^{-K} m}{4 + 3Kk/w} \right]$$

if A is the left-justifying algorithm.

Remark. When m and w are large, the bound becomes $\{1 + 0.75[\log_2 k]\}\, m$ approximately, a form that is easier to compare with Theorem 1.

Proof. Let w' denote $2^K[2^{-K}w]$, the largest multiple of 2^K not exceeding w. Also define

$$m' = 2^{K+2} \left[\frac{2^{-K-2} mw}{w + 0.75K(w - w')} \right] > 2^{K+2} \left[\frac{2^{-K} m}{4 + 3Kk/w} \right]$$

(the inequality follows because $w - w' < 2^K \leqslant k$). Now partition the bin into a *sub-bin*, consisting of the first w' columns, and a *remainder* of $w - w'$ columns. Since 2^K divides w' and 2^{K+2} divides m', Theorem 1 shows that

squares of sizes $1, 2, 4, \ldots, 2^K$ can be packed to level $(1 + 0.75K)\,m'$ in the sub-bin without using more total area than $m'w'$. This packing can be modified to a packing of the entire bin. The arrivals A_a need only fill r_a complete rows of the entire bin, the same number of rows as in packing the sub-bin. Since 2^a divides w', packed squares lie either entirely in the sub-bin or entirely in the remainder. Squares in the remainder do not depart at B_a. The total area packed remains less than

$$m'w' + (1 + 0.75K)\,m'(w - w') = m'\{w + 0.75K(w - w')\} \leqslant mw,$$

as required. The maximum level $(1 + 0.75K)\,m'$ is also as large as the corollary states. ∎

Apart from the left-justifying algorithm, these constructions can be modified for all first-fit algorithms. The main effect is to weaken the term $0.75 \log_2 k$ of Theorem 1 to $0.8[\log_4 k] \leqslant 0.4 \log_2 k$.

THEOREM 2. *Let A be any first-fit algorithm and k any integer. For large values of m and w, lists L of squares no larger than k by k can achieve levels*

$$A(L) \geqslant m\{1 + 0.8[\log_4 k]\}\{1 - O(1/m) - O(1/w)\}.$$

Proof. Let K denote $[\log_4 k]$. Use squares of sizes $s_a = 4^a$ for $0 \leqslant a \leqslant K$. Again choose the number of arrivals at A_a so that only complete rows of s_a-squares are packed. Also choose the number r_a of rows as large as possible subject to two conditions: (i) the total area packed must not exceed mw and (ii) 4^{K-a} must divide r_a. After A_a, the total packed area is at least $mw - 4^{K-a}4^a w \geqslant (m - k)\,w$. For, with any smaller packed area, 4^{K-a} more complete 4^a-rows could be packed without violating the conditions. By the same argument, if area S_a departs at B_a, then the area of the 4^{a+1}-rows packed at A_{a+1} is at least $S_a - kw$.

After A_0 there are $r_0 \geqslant m - k$ complete rows of 1-squares. Let B_0 remove all but rows 4, 8, 12, 16,..., and alternate squares within these remaining rows. The packing continues, somewhat as in Theorem 1. After A_a the band of r_i rows of s_i-squares is partitioned into strips of 4^{a-i} consecutive rows; all but the top row will be empty. As illustrated in Fig. 2, these strips are combined, now in quadruplets instead of pairs. During B_a the s_i-squares in the three lower strips depart and alternate s_i-squares in the top row of the upper strip depart. The departing area S_a depends on just how A packed the rows. We now show that, to within a term $O(1/w)$, at least $\frac{4}{5}$ of the area of each quadruplet departs at B_a. When originally packed a complete s_i-row might have contained as many as $[w/s_i] \leqslant w/s_i$ squares or as few as $(w - s_i + 1)/(2s_i - 1)$. The latter number is at least $\frac{1}{2}w/s_i$ when

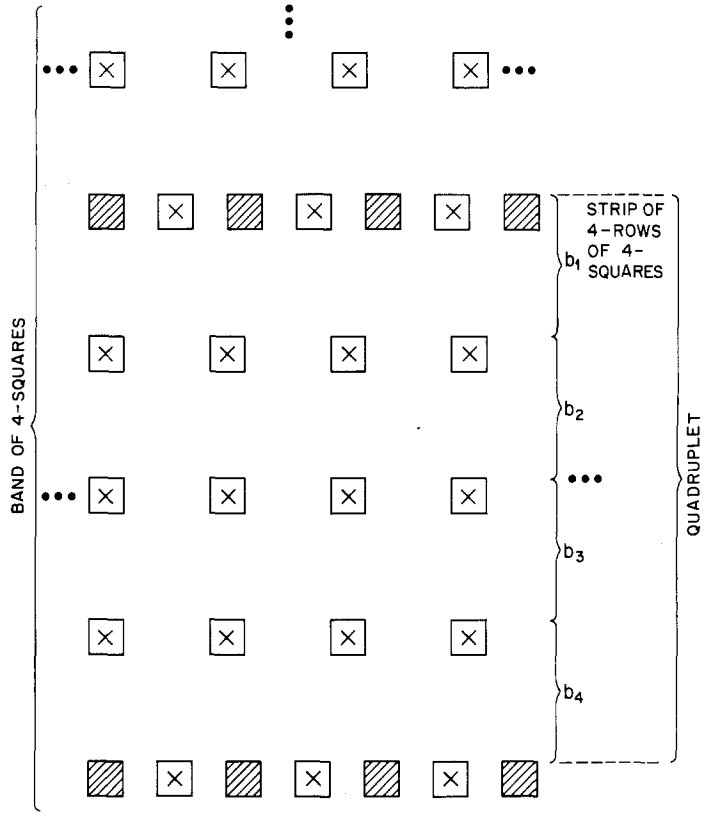

FIG. 2. The band of 4-squares after packing 16-squares. Squares with X's leave before 64-squares arrive. The departing area fraction is near $\frac{4}{5}$.

$w \geqslant 2k(k-1)$. Thus, a complete s_i-row cannot have less than $\frac{1}{2}$ the area of any other complete s_i-row (to within a term $O(1/w)$). Now let b_1, b_2, b_3, and b_4 be the areas of the 4 strips in a quadruplet, with b_1 referring to the top strip as in Fig. 2. By the definition of B_a, all of b_2, b_3, and b_4 are removed and within $O(1/w)$ of $\frac{1}{2}$ of b_1 is removed. Thus, the fraction of area removed is within $O(1/w)$ of

$$\frac{\frac{1}{2}b_1 + b_2 + b_3 + b_4}{b_1 + b_2 + b_3 + b_4} = 1 - \frac{b_1/2}{b_1 + b_2 + b_3 + b_4} \geqslant \frac{4}{5},$$

the inequality following from $b_i \geqslant b_1/2$, $i = 2, 3, 4$ (Fig. 2 is a worst case).

Therefore $S_a \geqslant 0.8mw\{1 - O(1/w) - O(1/m)\}$. The remaining squares are separated by gaps of less than 4^{a+1} and so the s_{a+1}-squares of A_{a+1} all go into a higher band. The final packing consists of $m\{1 - O(1/m)\}$ partially

filled rows of 1-squares and $K = [\log_4 k]$ other bands of heights at least $0.8m\{1 - O(1/m) - O(1/w)\}$ each. ∎

Another generalization of Theorem 1 is to spaces of higher dimension d. The base of the bin is now a $(d-1)$-dimensional rectangular parallelepiped of sides $w_1, w_2, \ldots, w_d - 1$. If k is a power of 2 dividing $w_1, w_2, \ldots, w_{d-1}$ and $m/4$, and if A is left-justifying, then lists exist with

$$A(L) = m\{1 - 2^{-d}) \log_2 k\}.$$

Section 6 will give more complicated lists that improve on Theorems 1 and 2.

4. An Upper Bound

When m, w, and k are large, Theorem 2 shows that $A(L)$ can be as large as $O(m \log k)$ for all first-fit algorithms. This section extends a result of Robson [7] to an upper bound on $A(L)$ for all lists L. When $k \to \infty$ and m grows as fast as k^2 this bound will also be $O(m \log k)$.

THEOREM 3. *Let squares of L be no larger than k by k. Given $w \geqslant 2k$, and any first-fit algorithm A,*

$$A(L) \leqslant \frac{H_k}{\ln 2 - 1/2} m' + (3k+1)\, k/2, \tag{1}$$

where $H_k = 1 + 1/2 + \cdots + 1/k$ and $m' = mw/(w - 2k)$.

Proof. Any j by j square of cells wil be called a j-region. A j-region, although square, need not be a j-square; some of its cells may be unoccupied. Let $a(Y)$ denote the area of set Y.

LEMMA 1. *For some j and $i \leqslant j+1$ let Y be a $(j+i)$-region within some packing. Suppose that only packed squares of size $i \times i$ or larger have nonzero area of intersection with Y. If a $(j+1)$-square cannot be packed into Y, then packed squares occupy at least i^2 cells of Y.*

Proof. Exactly i^2 different $(j+1)$-regions are subsets of Y (their upper right-hand cells lie in an $i \times i$ square). Since no $(j+1)$-square can be packed in Y, each $(j+1)$-region in Y must be *blocked*; i.e., must contain at least one cell occupied by some packed square S. Now $a(S \cap Y)$ will be proved to be an upper bound on the number of $(j+1)$-regions that S blocks.

S can overlap Y in three different ways. First, S may lie entirely in Y. Since S is assumed to be of size $i \times i$ or more, S can be seen to block all i^2

of the $(j+1)$-regions. But also $i^2 \leqslant a(S \cap Y)$. Second, S may overlap a corner of Y, say the lower left corner. Then the regions blocked by S all have their lower left corner cell in $S \cap Y$, and so number $a(S \cap Y)$ at most. Third, S may overlap an edge of Y, but no corner. Suppose S overlaps the first s columns at the left edge of Y. Since S occupies at least i cells in each of the s columns, S blocks all $(j+1)$-regions having leftmost column overlapping S. The number blocked is $i \min(s, i) \leqslant a(S \cap Y)$.

The lemma now follows: The area occupied in Y exceeds the number of $(j+1)$-regions blocked. But, all $i^2(j+1)$-regions are blocked. ∎

To prove Theorem 3 let D_j, $1 \leqslant j \leqslant k$, denote the maximum height ever achieved by a j-square. We shall prove by induction on j that

$$D_j \leqslant \frac{H_j}{\ln 2 - 1/2} m' + \sum_{i=1}^{j} (k+i).$$

Since $A(L) = \max_{1 \leqslant j \leqslant k} D_j$ and $\sum_{i=1}^{j} (k+i) = (2k+j+1)\, j/2 \leqslant (3k+1)\, k/2$, $1 \leqslant j \leqslant k$, the theorem will follow directly. The inequality with $j=1$ is trivial because $D_1 \leqslant m$ and $[\ln 2 - 1/2]^{-1} = 5.1773984\ldots$.

Introduce $c = [\ln 2 - 1/2]^{-1}$, $b_i = \sum_{l=1}^{i} (k+l)$, $i \geqslant 1$, and define $D_0 = b_0 = H_0 = 0$. We suppose $D_i \leqslant cH_i m' + b_i$, $1 \leqslant i \leqslant j$, and must prove that $D_{j+1} \leqslant cH_{j+1} m' + b_{j+1}$. Thus consider the packing produced by A just after a $(j+1)$-square has reached height D_{j+1}.

Consider the region X_i of all cells at levels $D_{i-1}+1, D_{i-1}+2, \ldots, D_i$. Since X_i lies entirely above level D_{i-1}, packed squares that intersect X_i have size $i \times i$ or larger. Lemma 1 shows that any $(j+1)$-region, lying entirely in X_i, contains at least i^2 occupied cells. Since X_i is a rectangle of height $D_i - D_{i-1}$ and width w, one can find $[(D_i - D_{i-1})/(j+i)][w/(j+i)]$ disjoint $(j+i)$-regions in X_i. Then the number of occupied cells in X_i is at least

$$\begin{aligned}
&[(D_i - D_{i-1})/(j+i)][w/(j+i)]\, i^2 \\
&\quad \geqslant \left\{ \frac{D_i - D_{i-1} + 1}{j+i} - 1 \right\} \left\{ \frac{w+1}{j+i} - 1 \right\} i^2 \\
&\quad \geqslant (D_i - D_{i-1} - j - i)(w - 2k) \left(\frac{i}{j+i} \right)^2. \qquad (2)
\end{aligned}$$

With $i = j+1$, $(j+1)$-squares are blocked only in rows up to row $D_{j+1} - 1$; however, the same steps with little change lead again to (2), with i replaced by $j+1$.

Since the occupied area in $X_1, X_2, \ldots, X_{j+1}$ is at most $mw = m'(w-2k)$, we have

$$\sum_{i=1}^{j+1} \left(\frac{i}{j+i} \right)^2 (D_i - D_{i-1} - j - i) \leqslant m'. \qquad (3)$$

Now write $D_i = cm'H_i + b_i - \delta_i$. Clearly, $\delta_0 = 0, \delta_1 > 0$ and we need only prove that $\delta_{j+1} \geqslant 0$ if $\delta_i \geqslant 0$ for $0 \leqslant i \leqslant j$. Inequality (3) now becomes

$$cm' \sum_{i=1}^{j+1} \frac{i}{(j+i)^2} + \sum_{i=1}^{j+1} \frac{i^2(k+i)}{(j+i)^2} - \sum_{i=1}^{j+1} \left(\frac{i}{j+i}\right)^2 (\delta_i - \delta_{i-1}) \leqslant m' + \sum_{i=1}^{j+1} \frac{i^2}{j+i}.$$

The second sum on the left and the sum on the right may be removed because $k \leqslant j$. The sum containing $(\delta_i - \delta_{i-1})$ may be rearranged into

$$\left(\frac{j+1}{2j+1}\right)^2 \delta_{j+1} - \sum_{i=1}^{j} \left\{ \left(\frac{i+1}{j+i+1}\right)^2 - \left(\frac{i}{j+i}\right)^2 \right\} \delta_i \geqslant \left(\frac{j+1}{2j+1}\right)^2 \delta_{j+1},$$

the inequality following because $(i/(j+i))^2 = (1 - j/(j+i))^2$ is an increasing function of i and because $\delta_i \geqslant 0$. These simplifications reduce (3) to

$$\left(\frac{j+1}{2j+1}\right)^2 \delta_{j+1} \geqslant \left\{ c \sum_{i=1}^{j+1} \frac{i}{(j+i)^2} - 1 \right\} m'. \tag{4}$$

The real function $x/(j+x)^2$ is monotone increasing in $0 \leqslant x \leqslant j$, and hence

$$i/(j+i)^2 = \int_{i-1}^{i} \frac{idx}{(j+x)^2} \geqslant \int_{i-1}^{i} \frac{xdx}{(j+x)^2},$$

and

$$\sum_{i=1}^{j+1} \frac{i}{(j+i)^2} \geqslant \int_0^j \frac{xdx}{(j+x)^2} = \ln 2 - \frac{1}{2} = 1/c.$$

Then $\delta_{j+1} \geqslant 0$ follows immediately from (4). ∎

COROLLARY 1. *Consider any first-fit algorithm A and all lists L having a fixed maximum square-size $k \times k$. Define $R_k = \limsup_{m \to \infty, w \to \infty} \sup_{A,L} A(L)/m$. Then*

$$0.2 + 0.4 \log_2 k \leqslant R_k \leqslant \frac{H_k}{\ln 2 - 1/2}. \tag{5}$$

These two bounds, which follow from Theorems 2 and 3, differ by a factor near $\ln 2/\{0.4(\ln 2 - 1/2)\} = 8.97....$ Sections 5 and 6 will suggest that the factor can be reduced to about 1.7.

Theorem 3 generalizes to a bin of sides $w_1,\ldots, w_{d-1}$, h in space of dimension $d \geqslant 2$. Suppose $w_i \geqslant 2k$ for $i = 1,\ldots, d-1$, and write

$$m' = m \prod_{i=1}^{d-1} \left(\frac{w_i}{w_i - 2k} \right).$$

Then Theorem 3 holds again with (1) replaced by

$$A(L) \leqslant c(d)\, H_k m' + (3k+1)\, k/2, \tag{1'}$$

where

$$c(d) = \left\{ \ln 2 - \sum_{i=1}^{d-1} 2^{-i}/i \right\}^{-1}.$$

The one-dimensional form of (1'), derived by Robson [7], required no term $(3k+1)\,k/2$.

For $d = 2, 3, 4, 5$, $c(d) = 5.177,\ 14.67,\ 37.64,\ 92.12$. This coefficient appears in the generalized form of Corollary 1, in which (5) becomes

$$1 + (1 - 2^{-d}) \log_2 k \leqslant R_k \leqslant c(d)\, H_k. \tag{5'}$$

Although $R_k = O(H_k) = O(\log k)$ for large k in every dimension d, the bounds on R_k get further apart in higher dimensions.

5. An Improved Upper Bound

The terms of (3) may be rearranged to obtain

$$\left(\frac{j+1}{2j+1} \right)^2 D_{j+1} \leqslant m' + \sum_{i=1}^{j+1} \frac{i^2}{j+1} \left\{ \left(\frac{i+1}{j+i+1} \right)^2 - \left(\frac{i}{j+i} \right)^2 \right\} D_i. \tag{6}$$

Since the coefficient of D_i is positive, (6) remains true with D_i replaced by an upper bound. Thus, by making it an equality, (6) becomes a recurrence for an upper bound D'_j on D_j. From the form of (6), D'_k will be a linear function of m':

$$D'_k = \alpha_k m' + \beta_k; \tag{7}$$

for instance $D'_1 = m' + 1$ and $D'_2 = (43m' + 73)/16$. Coefficients α_k, β_k with larger k appear in Table I. Theorem 3 also gave a linear bound, differing from (7) mainly in having cH_k in place of α_k. The last column of Table I shows (7) to be a better bound, particularly when k is not large. The constant term β_k in (7) is also less than the corresponding term of (1).

Since D_{j+1} is an integer, (6) can also be improved by multiplying through

TABLE I

Coefficients of the Bound $D'_k = \alpha_k m' + \beta_k$

Square Size k	α_k	β_k	$\alpha_k/cH_k)$
1	1	1	0.193
2	2.688	4.56	0.346
3	3.985	10.48	0.420
4	5.036	18.82	0.467
6	6.670	42.78	0.526
8	7.917	76.44	0.563
10	8.923	119.82	0.588
16	11.127	308.29	0.636
20	12.205	482.55	0.655
32	14.521	1238.71	0.691
64	18.010	4966.69	0.733
128	21.549	19891.53	0.766

by $(2j+1)^2/(j+1)^2$ and taking the integer part of the right-hand side. The resulting inequality can also be used as a recurrence for a bound which, however, is only very slightly better than (7).

6. An Improved Lower Bound

Theorem 1 failed to raise k-squares as high as possible for two main reasons. One is that other first-fit algorithms could have attained higher levels with the same list L. For instance, the twelve 4-squares in Fig. 1 could have been separated by gaps of up to 3 units within each row. There could have been 6 rows of two 4-squares; D_4 would then have been 52 instead of 40.

A second way to raise k-squares to a higher level is to pack squares of all sizes $1, 2, \ldots, k$ instead of just powers of 2. As in Section 3, L will specify blocks of arrivals and departures $A_0, B_0, A_1, \ldots, A_k$. At B_{j-1} the packed squares are visited in order of level and from left to right. Each visited square departs if its removal does not make room for an s_j-square. At A_j as many s_j-squares as possible arrive without violating the limit mw on area. Figure 3 improves the example with $w = m = 16$, $k = 4$, and left-justifying A_j of Fig. 1. Now B_1 removes 2-squares but not 1-squares. D_4 increases from 40 to 49. A first-fit algorithm that packs each row as loosely as possible would improve D_4 further to 57. Even this value is not very close to the upper bound $D'_4 = 179$ from (7). Closer agreements will be obtained when, for fixed k, both m and w become large numbers.

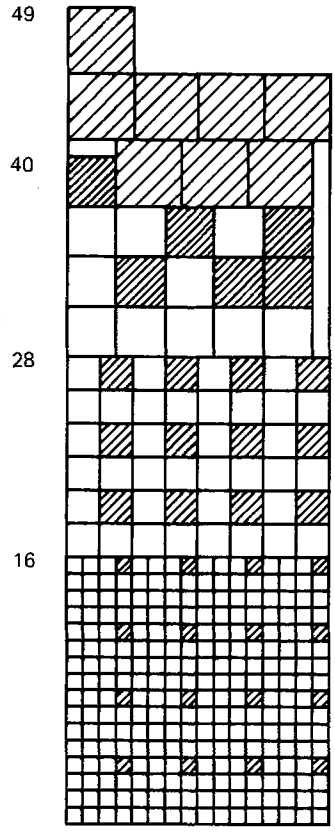

FIG. 3. Packing with $w = m = 16$ and $k = 4$, for comparison with Fig. 1.

Let M denote the smaller of m and w. In the limit as $M \to \infty$, lists L will be found for which

$$D_k = \gamma_k\{1 + O(1/M)\}\, m. \tag{8}$$

Here, γ_k is a coefficient depending only on k. The bounds on R_k of Corollary 1 are thereby improved to

$$\gamma_k \leqslant R_k \leqslant \alpha_k. \tag{9}$$

The L that lead to (8) will avoid configurations like Fig. 3, in which packed squares of different sizes share the same row. Let only complete rows of j-squares arrive. If area $\alpha_j mw$ is available, the left-justifying rule will pack j-squares in rows, each containing a number, $\rho = [w/j]$, of j-squares. There

will be $[a_j mw/(\rho j^2)]$ rows and $\rho[a_j mw/(\rho j^2)]$ arrivals. As $M \to \infty$, $\rho = (w/j)\{1 + O(1/M)\}$, and hence

$$D_j - D_{j-1} = a_j m\{1 + O(1/M)\}. \tag{10}$$

Similarly, the algorithm that packs each row as loosely as possible (with j-squares spaced $j-1$ units apart), packs $\rho = [(w+j-1)/(2j-1)]$ j-squares per row. Then (10) becomes

$$D_j - D_{j-1} = (2 - j^{-1})\, a_j m\{1 + O(1/M)\}. \tag{11}$$

It remains to show that, in the procedure for constructing L, there is indeed a constant a_j such that the j-squares arrive with total area $a_j mw\{1 + O(1/M)\}$. Then the recurrence (10) or (11) holds and the solution has the desired form (8). The induction argument that follows will show for certain numbers $a_i(j)$ independent of m and w, that the i-squares, still packed after j-squares arrive, have total area $a_i(j)\, mw\{1 + O(1/M)\}$. Of course, then $a_j = a_j(j)$.

Consider first the left-justifying algorithm A. Add to the induction hypothesis the statement that, right after the j-squares are packed, the i-squares are in a square-lattice configuration; let $x_i(j)$ denote the separation between i-squares. The procedure begins with available area mw and packs 1-squares with $a_1(1) = a_1 = 1$ and spacing $x_1(1) = 0$. Suppose the induction hypothesis is true for $1, 2, \ldots, j-1$ and consider B_{j-1}. If

$$2x_i(j-1) + i \geqslant j \tag{12}$$

then no i-square departs, $x_i(j) = x_i(j-1)$, and $a_i(j) = a_i(j-1)$. If (12) fails, then the procedure removes alternate rows and columns of i-squares; i.e., $x_i(j) = 2x_i(j-1) + i$ and $a_i(j) = a_i(j-1)/4$. Thus $a_j(j) = \Sigma^* 3a_i(j-1)/4$

TABLE II

Level Attained in Packing the List L of Section 6 (Large m and w)

		$\gamma_k = level/m$		
k	$1 + 0.75 \log_2 k$	Justified A	Loose A	α_k
1	1	1	1	1
2	1.75	1.75	2.125	2.688
4	2.5	2.922	3.406	5.036
8	3.25	4.465	5.217	7.917
16	4	6.233	7.031	11.127
32	4.75	8.117	8.917	14.521
64	5.5	10.053	10.875	18.010

with Σ^* denoting a sum over all i for which (12) fails. Although $a_i(j)$, $x_i(j)$, and γ_k have no simple explicit formulas, they can be calculated from the rules following (12).

When A is changed to the loose-packing algorithm i-squares become arranged in a rectangular lattice requiring different horizontal and vertical spacings $x_i(j)$ and $y_i(j)$. The details will be omitted. Numerical results in Table II improve Theorem 1 considerably; γ_k and α_k are reasonably close when k is not too large. In one dimension, similar arguments improve Robson's results. For example, with $k = 64$, Robson's bounds differ by a factor 1.71 while $\alpha_{64}/\gamma_{64} = 1.31$.

RECEIVED: May 17, 1983; ACCEPTED: June 12, 1984

REFERENCES

1. COFFMAN, E.G., JR. (1983), An introduction to combinatorial models of dynamic storage allocation, *SIAM Rev.* **25**, 311–325.
2. COFFMAN, E.G., JR., GAREY, M.R., AND JOHNSON, D.S. (1983), Dynamic bin-packing, *SIAM J. Comput.* **12**, 227–258.
3. KNUTH, D.E. (1973), "Fundamental Algorithms," Vol. 1, 2nd ed., Addison-Wesley, Reading, Mass.
4. MCILROY, M.D. (1968), "Some Problems in Dynamic Storage Allocation," Bell Labs, Murray Hill, NJ, unpublished typescript.
5. ROBSON, J.M. (1971), An estimate of the store size necessary for dynamic storage allocation, *J. Assoc. Comput. Mach.* **18**, 416–423.
6. ROBSON, J.M. (1974), Bounds for some functions concerning dynamic storage allocation, *J. Assoc. Comput. Mach.* **21**, 491–499.
7. ROBSON, J.M. (1977), Worst-case fragmentation of first-fit and best-fit storage allocation strategies, *Comput. J.* **20**, 242–244.